# Báo cáo kết quả thực nghiệm của Trần Vỹ Khang. MSSV 22520628

https://github.com/khang2204/SortAlgorithms

**Merge sort**

**Ý tưởng**

Sắp xếp trộn hoạt động kiểu đệ quy:

- Đầu tiên chia dữ liệu thành 2 phần, và sắp xếp từng phần.
- Sau đó gộp 2 phần lại với nhau. Để gộp 2 phần, ta làm như sau:
  - Tạo một dãy A mới để chứa các phần tử đã sắp xếp.
  - So sánh 2 phần tử đầu tiên của 2 phần. Phần tử nhỏ hơn ta cho vào A và xóa khỏi phần tương ứng.
  - Tiếp tục như vậy đến khi ta cho hết các phần tử vào dãy A.

**Ưu điểm**

- Chạy nhanh, độ phức tạp O(N∗logN)
- Ổn định

**Nhược điểm**

- Cần dùng thêm bộ nhớ để lưu mảng A.

**Sắp xếp vun đống (HeapSort)**

**Ý tưởng**

Ta lưu mảng vào CTDL Heap.

Ở mỗi bước, ta lấy ra phần tử nhỏ nhất trong heap, cho vào mảng đã sắp xếp.

**Ưu điểm**

- Cài đặt đơn giản nếu đã có sẵn thư viện Heap.
- Chạy nhanh, độ phức tạp O(N∗logN)

**Nhược điểm**

- Không ổn định

**Sắp xếp nhanh (QuickSort)**

**Ý tưởng**

- Chia dãy thành 2 phần, một phần "lớn" và một phần "nhỏ".
  - Chọn một khóa **pivot**
  - Những phần tử lớn hơn **pivot** chia vào phần lớn

    - Những phần tử nhỏ hơn hoặc bằng **pivot** chia vào phần nhỏ.
- Gọi đệ quy để sắp xếp 2 phần.

**Ưu điểm**

- Chạy nhanh (nhanh nhất trong các thuật toán sắp xếp dựa trên việc só sánh các phần tử). Do đó quicksort được sử dụng trong nhiều thư viện của các ngôn ngữ như Java, C++ (hàm `sort` của C++ dùng Intro sort, là kết hợp của Quicksort và Insertion Sort).

**Nhược điểm**

- Tùy thuộc vào cách chia thành 2 phần, nếu chia không tốt, độ phức tạp trong trường hợp xấu nhất có thể là O(N^2). Nếu ta chọn pivot ngẫu nhiên, thuật toán chạy với độ phức tạp trung bình là O(N∗logN) (trong trường hợp xấu nhất vẫn là O(N^2)nhưng ta sẽ không bao giờ gặp phải trường hợp đó).
- Không ổn định.

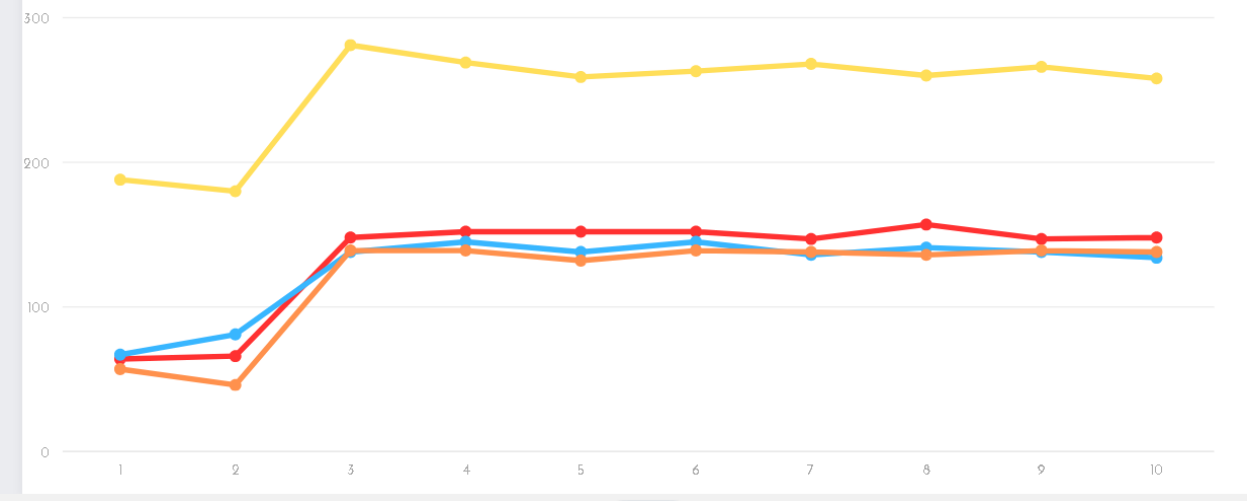
**Thực nghiệm đánh giá với 10 bộ test case được sinh ngẫu nhiên**: bộ dữ liệu gồm 10 dãy, mỗi dãy khoảng 1 triệu số thực (ngẫu nhiên); dãy thứ nhất đã có thứ tự tăng dần, dãy thứ hai có thứ tự giảm dần, 8 dãy còn lại trật tự ngẫu nhiên

**Bằng 1 đoạn code gồm 4 thuật toán:** Quick Sort, Heap Sort, Merge Sort, std::sort()

**Kết quả:**

| Thời gian chạy các thuật toán sắp xếp đơn vị ms | | | | |
|---|---|---|---|---|
| | MergeSort | QuickSort | HeapSort | std::sort in C++ |
| 1 | 64 | 67 | 188 | 57 |
| 2 | 66 | 81 | 180 | 46 |
| 3 | 148 | 138 | 281 | 139 |
| 4 | 152 | 145 | 269 | 139 |
| 5 | 152 | 138 | 259 | 132 |
| 6 | 152 | 145 | 263 | 139 |
| 7 | 147 | 136 | 268 | 138 |
| 8 | 157 | 141 | 260 | 136 |
| 9 | 147 | 138 | 266 | 139 |
| 10 | 148 | 134 | 258 | 138 |

Biểu đồ:




Màu đỏ: MergeSort

Màu xanh: QuickSort

Màu vàng: HeapSort

Màu cam: std::sort in C++

**Nhận xét:**

- Thuật toán Heap Sort có thời gian chạy lâu nhất trong 4 thuật toán do chi phí hằng số lớn. Khi mảng sắp xếp tăng dần thì việc xây dừng cấu trúc dữ liệu Heap trở nên tốn thời gian hơn do các phần tử trong Heap liên tục bị đổi gốc. Ngược lại khi dãy sắp xếp giảm dần thì thuật toán chạy hiêu quả và nhanh hơn.

- Hàm sort của thư viện chuẩn C++ có tốc độ chạy nhanh nhất bởi lẽ đã được tối ưu bằng việc kết hợp nhiều thuật toán khác nhau.

-Merge Sort và Quick Sort có thời gian chạy tương đối nhanh, Quick Sort chọn khóa ngẫu nhiên nên không ổn định bằng Merge Sort

**Kết luận:**

Thông qua việc thực nghiệm ta có thể biết được cơ chế và thời gian chạy của các thuật toán sắp xếp. Qua đó tùy theo mục đích sử dụng ta sẽ cài đặt thuật toán phù hợp với chương trình